UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,560	06/29/2000	Shekhar N. Swamy	MS147163.1	8517

27195 7590 02/05/2004

AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114

EXAMINER	
KISS, ERIC B	

ART UNIT	PAPER NUMBER
2122	15

DATE MAILED: 02/05/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PR9

Office Action Summary	Application No.	Applicant(s)	
	09/607,560	SWAMY ET AL.	
	Examiner	Art Unit	
	Eric B. Kiss	2122	

-- The MAILING DATE of this communication appears on the cover sh et with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 November 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24 and 26-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24 and 26-30* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *29 June 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____.
4) ☐ Interview Summary (PTO-413) Paper No(s). _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other:

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on November 26, 2003, has been entered. Claims 1-24 and 26-30 are pending.

Response to Arguments

2. Claims 3-15, 22, and 23 were previously rejected under 35 U.S.C. §112, second paragraph based on Applicant's use of the term "loop" in a manner apparently repugnant to its accepted meaning in the context of the relative arts (flow-graph-based computer program representations). This rejection has been withdrawn for reasons stated below.

The Examiner had previously presented Applicant with an accepted meaning for the term "loop" (as originally presented in the Office action mailed February 18, 2003) as follows:

> A collection of nodes in a flow graph such that 1) All nodes in the collection are *strongly connected*; that is, from any node in the loop to any other, there is a path of length one or more, wholly within the loop, and 2) The collection of nodes has a unique *entry*, that is, a node in the loop such that the only way to reach a node of the loop from a node outside the loop is to first go through the entry. [see pages 533-534 of Alfred V. Aho et al. "Compilers: Principles, Techniques, and Tools," 1986, Addison-Wesley]

Applicant had traversed the rejection by providing an alternate definition of "loop". In Applicant's reply filed June 16, 2003, Applicant had stated the following:

> The term "loop" as employed by the applicants is not repugnant to an alternative meaning of the term "loop" as used in the art. The term "loop" when employed as a verb means to "move in loops or in an arc," *Merriam-Webster Dictionary,* http://www.m-w.com/cgi-bin/dictionary. The specification at page 12 lines 7-11 states that "a source loop path node may be used in a subsequent code generation pass... as the point around which the generated code will... loop." The verb "loop" found at the end of such sentence is referring to the term "source loop path node," thus rendering the two consistent. A "loop point" is defined in the specification at page 12 lines 17-19 as "a source tree node for which maxoccurs = *." The highest-level "loop point" (*e.g.*, a loop point furthest away from its root node) becomes the "source loop path node," and thus generated code will "loop" around that particular node.

In response, the Examiner noted the following flaws in the above argument (as presented in the Office action mailed July 2, 2003):

> a) Applicant cites a definition of "loop" as a verb whereas the claimed terminology in question is a noun/adjective.
> b) Applicant's cited definition is circular, i.e., the term "loops" appears in the cited definition of "loop".
> c) The URL cited by the Applicant does not, as of the time of this writing, point to a document containing the cited definition or to links making such definition accessible.
> d) The Applicant's cited definition appears to be less relevant to the art to which the claimed subject matter pertains than the Examiner-cited definition.

In Applicant's reply filed September 17, 2003, Applicant redirected the interpretation of the term "loop" in the following statements:

> To clarify, one aspect of the subject invention relates to a method of generating code from a defined mapping between a source schema and a target schema. Accordingly, "loop" should be interpreted relative to the manner in which such term is employed in the computer programming and/or coding arts. In computer science it is commonly known that a loop is a block of code that repeats until a condition is satisfied (*e.g.*, for, while, until...). For instance, an introductory programming book states that "A loop executes a sequence of statements until a particular condition is true (or false)" (Ivor Horton, "Beginning Visual C++ 6",

> 1998, Wrox Press Ltd., page 110) (attached hereinafter as Exhibit #1). Claims 3-15, 22, and 23 refer to a source loop path node and a source loop path. The specification specifically states "A loop point or source loop path node is a source tree node for which maxoccurs = *" (page 12, lines 17-18). When maxoccurs = * more than one occurrence of a node is possible (page 12, lines 2-4), thus it is necessary to generate looping code (*e.g.*, for each node...do something) to ensure that all nodes and node data are properly captured. Accordingly, applicants' use of the term "loop" is consistent within the context of the relevant art, namely computer programming.

Initially, the Examiner submits that Applicant's most recently applied definition of "loop", namely a block of code that repeats until a condition is satisfied, can be considered as one valid definition within the field of Computer Science, specifically within the area of high-level computer programming language (source code) syntax. Upon further consideration of the relative context of Applicant's use of the term "loop", it appears that although the terms "source loop path" and "source loop path node" are used within the context of a flow-graph-based code generation (seemingly validating the application of the Examiner's provided definition of loop), the specification does apparently describe the generation of looping code (source code) based on the graphs. For example, in the instant specification, on page 12, in lines 7-11, Applicant discloses:

> The source loop path node may be used in a subsequent code generation pass (e.g., pass 4 (250) as illustrated in Figs. 9A-9D and described in greater detail infra) as the point around which the generated code will iterate or loop. As an example, where the generated code is XSL, the source loop path node may be used in generating <xsl : for-each> code.

Based on the above-cited portion and other relevant portions of the specification, the Examiner concedes that Applicant's use of the term "loop" is appropriate and in agreement with

the provided definition of loop (a block of code that repeats until a condition is satisfied).

Accordingly, the previous rejection under 35 U.S.C. §112, second paragraph, is withdrawn.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 1-24 and 26-30 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claims 1, 20, 29, and 30 recite "wherein the mapping comprises generically [specified/defined] data transformations from the source schema to the target schema". The instant specification does not provide any definition of "generically" specified/defined data transformations, and Applicant's arguments accompanying this newly added feature describe only an example of such a mapping ("e.g. visual representation of a map created by a user"; see, for example, Applicant's reply filed November 3, 2003 (Paper No. 10), p. 13, lines 9-10). Since the meaning of generically specified transformations is not conveyed by the specification with reasonable clarity, one of ordinary skill in the art would not be able to practice or make the claimed invention.

The Examiner notes that the instant specification does support "graphically" specified data transformations from the source schema to the target schema (see, for example, p. 20, lines 11-19). In the interest of compact prosecution, the Examiner interprets the word "generically" in the limitation "wherein the mapping comprises generically [specified/defined] data transformations from the source schema to the target schema" as if it reads –graphically– for the purpose of further examination.

Claim Rejections - 35 USC § 102

5. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

6. Claims 1, 20, 21, 23, 25, 26, 29, and 30 are rejected under 35 U.S.C. 102(b) as being anticipated by Greger Lindén, "Structured Document Transformations," 1997, University of Helsinki, Finland, Series of Publications A, Report A-1997-2 (hereinafter *Lindén*).

As per claim 1, *Lindén* discloses a method of generating code representing a mapping between a source schema and a target schema, the mapping comprising graphically specified (see, for example, page 43, paragraph 1, "ALCHEMIST also provides a graphical interface for specifying transformations") data transformations from the source schema to the target schema, the source schema comprising a source tree having a source node and the target schema comprising a target tree having a target node (see "TT-grammars," description on pages 31-34

and illustrative example on pages 34-42), the method comprising: determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema (specifying production group associations and symbol associations; see pages 49-51; and page 32, second paragraph). As disclosed by Lindén (see, for example, pages 49-51), a mapping connects the source and target grammars together based on the TT-grammar technique. In generating the mapping, the user connects source and target subgrammars by forming production group associations and specifying symbol associations. As described on page 21, in paragraph 4, and in Algorithm 3.2 on pages 33-34, the production group associations are used to generate target subtrees by processing the corresponding source subtrees. Thus, a tracing through the mapping from target to source is necessary in order to process source subtrees corresponding to the target subtrees.

Lindén further discloses matching hierarchy by generating a hierarchy match list for the target node (constructing the corresponding target parse tree; see "source-to-target mapper" description on page 56). As disclosed by Lindén (see, for example, page 56), a target parse tree corresponding to the source parse tree is constructed according to the TT-grammar. The target parse tree is a hierarchical data structure containing target nodes that are matched (corresponding to) the hierarchical data structure of source parse tree nodes based on the TT-grammar and corresponding specified mapping.

Lindén further discloses generating code according to the hierarchy match list (see subsection 4.3.2 on pages 51-52). As disclosed by Lindén (see, for example, section 4.3.2 on pages 51-52), code is generated according to the mapping and target parse tree. Lindén discloses

specialized tools, such as STONE, which generates a mapper from the mapping specification, and SWINDLER, which generates an unparser from the target grammar.

As per claims 20 and 25, *Lindén* discloses a method for compiling a graphically specified (see, for example, page 43, paragraph 1, "ALCHEMIST also provides a graphical interface for specifying transformations") mapping between a source schema having source nodes associated therewith, and a target schema having target nodes associated therewith (see "TT-grammars," description on pages 31-34 and illustrative example on pages 34-42), comprising: determining source node dependencies for at least one target node by tracing from the at least one target node through the mapping to the source schema (specifying production group associations and symbol associations; see pages 49-51; and page 32, second paragraph). As disclosed by Lindén (see, for example, pages 49-51), a mapping connects the source and target grammars together based on the TT-grammar technique. In generating the mapping, the user connects source and target subgrammars by forming production group associations and specifying symbol associations. As described on page 21, in paragraph 4, and in Algorithm 3.2 on pages 33-34, the production group associations are used to generate target subtrees by processing the corresponding source subtrees. Thus, a tracing through the mapping from target to source is necessary in order to process source subtrees corresponding to the target subtrees.

Lindén further discloses matching hierarchy by generating a hierarchy match list for the target node (constructing the corresponding target parse tree; see "source-to-target mapper" description on page 56). As disclosed by Lindén (see, for example, page 56), a target parse tree corresponding to the source parse tree is constructed according to the TT-grammar. The target

parse tree is a hierarchical data structure containing target nodes that are matched (corresponding to) the hierarchical data structure of source parse tree nodes based on the TT-grammar and corresponding specified mapping.

Lindén further discloses generating code according to the hierarchy match list (see subsection 4.3.2 on pages 51-52). As disclosed by Lindén (see, for example, section 4.3.2 on pages 51-52), code is generated according to the mapping and target parse tree. Lindén discloses specialized tools, such as STONE, which generates a mapper from the mapping specification, and SWINDLER, which generates an unparser from the target grammar.

As per claims 21 and 23, *Lindén* further discloses generating a source dependency list (specifying production group associations and symbol associations; see pages 49-51; and page 32, second paragraph); and initializing node dependencies memory prior to determining source dependencies and later freeing that memory (inherent).

As per claim 26, *Lindén* further discloses matching hierarchy comprising top-down matching (see Algorithm 3.1 on pages 29-30).

As per claim 29, *Lindén* discloses a system for generating code representing a graphically specified (see, for example, page 43, paragraph 1, "ALCHEMIST also provides a graphical interface for specifying transformations") mapping between a source schema and a target schema, the mapping comprising data transformations from the source schema to the target schema, the source schema comprising a source tree having source nodes and the target schema comprising a target tree having a target node (see "TT-grammars," description on pages 31-34

and illustrative example on pages 34-42), the system comprising: means for determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema (specifying production group associations and symbol associations; see pages 49-51; and page 32, second paragraph). As disclosed by Lindén (see, for example, pages 49-51), a mapping connects the source and target grammars together based on the TT-grammar technique. In generating the mapping, the user connects source and target subgrammars by forming production group associations and specifying symbol associations. As described on page 21, in paragraph 4, and in Algorithm 3.2 on pages 33-34, the production group associations are used to generate target subtrees by processing the corresponding source subtrees. Thus, a tracing through the mapping from target to source is necessary in order to process source subtrees corresponding to the target subtrees.

Lindén further discloses means for matching hierarchy by generating a hierarchy match list for the target node (constructing the corresponding target parse tree; see "source-to-target mapper" description on page 56). As disclosed by Lindén (see, for example, page 56), a target parse tree corresponding to the source parse tree is constructed according to the TT-grammar. The target parse tree is a hierarchical data structure containing target nodes that are matched (corresponding to) the hierarchical data structure of source parse tree nodes based on the TT-grammar and corresponding specified mapping.

Lindén further discloses means for generating code according to the hierarchy match list (see subsection 4.3.2 on pages 51-52). As disclosed by Lindén (see, for example, section 4.3.2 on pages 51-52), code is generated according to the mapping and target parse tree. Lindén

discloses specialized tools, such as STONE, which generates a mapper from the mapping specification, and SWINDLER, which generates an unparser from the target grammar.

As per claim 30, *Lindén* discloses a computer-readable medium having computer-executable instructions for: generating code representing a graphically specified (see, for example, page 43, paragraph 1, "ALCHEMIST also provides a graphical interface for specifying transformations") mapping between a source schema and a target schema, the mapping comprising data transformations from the source schema to the target schema, the source schema comprising a source tree having a source node and the target schema comprising a target tree having a target node (see "TT-grammars," description on pages 31-34 and illustrative example on pages 34-42); determining source node dependencies for the target node by tracing from the target node through the mapping to the source schema (specifying production group associations and symbol associations; see pages 49-51; and page 32, second paragraph). As disclosed by Lindén (see, for example, pages 49-51), a mapping connects the source and target grammars together based on the TT-grammar technique. In generating the mapping, the user connects source and target subgrammars by forming production group associations and specifying symbol associations. As described on page 21, in paragraph 4, and in Algorithm 3.2 on pages 33-34, the production group associations are used to generate target subtrees by processing the corresponding source subtrees. Thus, a tracing through the mapping from target to source is necessary in order to process source subtrees corresponding to the target subtrees.

Lindén further discloses matching hierarchy by generating a hierarchy match list for the target node (constructing the corresponding target parse tree; see "source-to-target mapper"

description on page 56). As disclosed by Lindén (see, for example, page 56), a target parse tree corresponding to the source parse tree is constructed according to the TT-grammar. The target parse tree is a hierarchical data structure containing target nodes that are matched (corresponding to) the hierarchical data structure of source parse tree nodes based on the TT-grammar and corresponding specified mapping.

Lindén further discloses generating code according to the hierarchy match list (see subsection 4.3.2 on pages 51-52). As disclosed by Lindén (see, for example, section 4.3.2 on pages 51-52), code is generated according to the mapping and target parse tree. Lindén discloses specialized tools, such as STONE, which generates a mapper from the mapping specification, and SWINDLER, which generates an unparser from the target grammar.

Claim Rejections - 35 USC § 103

7. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

8. Claims 2 and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Lindén* in view of Alfred V. Aho et al. "Compilers: Principles, Techniques, and Tools," 1986, Addison-Wesley (hereinafter *Aho et al.*).

As per claim 2, *Lindén* discloses such a method (see disclosure applied above to claim 1) but fail to expressly disclose allocating memory for a compiler node; associating the compiler node with the target node; allocating memory for compiler variable classes; and associating

compiler variable classes with functoids. However, *Aho et al.* teach allocating memory for a compiler node (obtaining a block of storage); associating the compiler node with the target node (subdividing the storage to hold the generated target code); allocating memory for compiler variable classes (subdividing the storage to hold data objects); and associating compiler variable classes with functoids (keeping track of procedure activations; see "Subdivision of Run-Time Memory" on pages 396-397). Therefore, it would have been obvious to one having ordinary skill in the computer art at the time the invention was made to modify the method of *Lindén* to include memory initialization as per the teachings of *Aho et al.* One would be motivated to do so to establish a run-time environment.

As per claim 16, in addition to the disclosure and teachings applied above, *Lindén* further discloses generating a code header for a root node (pre-processing commands); generating a code trailer for the root node (post-processing commands); processing target record nodes in a preexecuteparent function, a postexecuteparent function, and an executeleaf function (processing the pre-processing and post-processing commands); and processing field nodes in the executeleaf function (processing normal spell commands; see the last paragraph on page 55 through the beginning of section 4.4 on page 57). Therefore, for reasons stated above, such a claim also would have been obvious.

9. Claims 18 and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over *Lindén* in view of Alberto Bellina, "XmlTool documentation," 21 January 2003 (hereinafter *Bellina*).

As per claims 18 and 28, *Lindén* discloses such a method (see disclosure applied to claims 1 and 28 above) but fails to expressly disclose creating an XSL style sheet representation

of the mapping. However, *Bellina* teaches a tool and method of manipulating XML schemas including generating XSL style sheet representations of mappings (see "XSL generator" on page 12). Therefore, it would have been obvious to one having ordinary skill in the computer art at the time the invention was made to modify the method of *Lindén* to include generation of XSL style sheet representations as per the teachings of *Bellina*. One would be motivated to do so to be able to produce output in a standard transformation language format for use with XML files.

10. Claims 27 is rejected under 35 U.S.C. 103(a) as being unpatentable over *Lindén*.

As per claim 27, *Lindén* discloses such a method (see disclosure applied above to claim 20) and furthermore discloses the transformation of SGML schemas (see section 2.4 on pages 20-23; and introduction to chapter 5 on page 59) but fails to expressly disclose the source schema and target schema being XML schemas. However, *Lindén* teaches the introduction of XML as a subset of SGML and suggests its future use as a replacement for SGML (see last paragraph on page 23). Therefore, it would have been obvious to one having ordinary skill in the computer art at the time the invention was made to modify the SGML schema transformation method of Lindén to include XML schemas as per Lindén's own suggestion. One would be motivated to do so to implement schemas that lack the drawbacks of full SGML.

Allowable Subject Matter

11. Claims 3-15, 17, 19, 22, and 24 would be allowable if rewritten to overcome the rejection(s) under 35 U.S.C. 112, first paragraph, set forth in this Office action and to include all of the limitations of the base claim and any intervening claims.

Conclusion

12. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Eric B. Kiss whose telephone number is (703) 305-7737. The Examiner can normally be reached on Tue. - Fri., 7:30 am - 5:00 pm. The Examiner can also be reached on alternate Mondays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Tuan Dam, can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

EBK/ebk
January 28, 2004

TUAN DAM
SUPERVISORY PATENT EXAMINER